II. AMENDMENTS TO THE CLAIMS

The following listing of claims replaces all prior versions, and listings, of claims in the application:

1. (Currently Amended) A ~~tree~~ graph walking system, comprising:

a binding system for binding a ~~tree~~ graph observer that looks for matching node patterns with a ~~tree~~ directed non-cyclic graph, for binding node patterns that identify distinguishing node attributes to node observers that at least one of analyze and process a particular node to generate at least one node pairing, and for binding the ~~tree~~ graph observer to at least one node pattern-node observer pairing;

~~tree~~ graph walking logic for systematically walking through nodes within the directed non-cyclic graph ~~tree~~;

a pattern testing system for determining if an attribute of an encountered node matches one of the node patterns;

an event manager for generating an encountered event when one of the node observers is bound to a matching node pattern; and

a pruning system that can deactivate the ~~tree~~ graph observer with respect to sub-nodes of the encountered node without deleting the sub-nodes if a bound node observer determines that there is no interest in the sub-nodes.

2. (Currently Amended) The ~~tree~~ graph walking system of claim 1, wherein the encountered event is handled by the bound node observer.

3. (Currently Amended) The ~~tree~~ graph walking system of claim 1, wherein the ~~tree~~ graph walking logic walks through the ~~tree~~ graph in a top down hierarchal manner.

4. (Currently Amended) The ~~tree~~ graph walking system of claim 1, wherein the pruning system can reactivate a deactivated ~~tree~~ graph observer after the sub-nodes of the encountered node have been walked.

5. (Currently Amended) The ~~tree~~ graph walking system of claim 1, wherein the event manager generates a completed event for each node observer that received an encountered event and that did not cause the ~~tree~~ graph observer to become deactivated.

6. (Currently Amended) The ~~tree~~ graph walking system of claim 5, wherein the completed event can cause the ~~tree~~ graph walking logic to repeat the walk through the sub-nodes.

7. (Currently Amended) The ~~tree~~ graph walking system of claim 1, wherein the pruning system can further cause the ~~tree~~ graph walking logic to bypass walking of the sub-nodes if the ~~tree~~ graph observer has been deactivated and no other active ~~tree~~ graph observers exist.

8. (Currently Amended) A system for analyzing a directed non-cyclic graph ~~tree~~ of hierarchical data, comprising:

a system for binding a plurality of ~~tree~~ graph observers that look for matching node patterns to a ~~tree~~ direced non-cyclic graph, wherein each ~~tree~~ graph observer is further bound to a set of node patterns that identify distinguishing node attributes and a set of node observers that at least one of analyze and process a particular node;

~~tree~~ graph walking logic for systematically walking through nodes within the ~~tree~~ graph;

a first pruning system that can be instructed by a node observer bound with an associated ~~tree~~ graph observer to deactivate the associated ~~tree~~ graph observer until a set of sub-nodes for the encountered node has been walked; and

a second pruning system that can instruct the ~~tree~~ graph walking logic not to walk the set of sub-nodes for the encountered node without deleting the set of sub-nodes.

9. (Currently Amended) The system of claim 8, wherein the second pruning system will cause the set of sub-nodes not to be walked only if all of the plurality of ~~tree~~ graph observers have been deactivated.

10. (Original) The system of claim 8, further comprising a pattern testing system for determining if the encountered node matches one of the node patterns.

11. (Original) The system of claim 8, further comprising an event manager for generating an encountered event when one of the node observers is bound to a matching node pattern.

12. (Currently Amended) A computer implemented method for analyzing a ~~tree~~ directed non-cyclic graph of hierarchical data, comprising the steps of:

binding a plurality of ~~tree~~ graph observers that look for matching node patterns to a ~~tree~~ directed non-cyclic graph, wherein each ~~tree~~ graph observer is further bound to a set of node patterns that identify distinguishing node attributes and a set of node observers that at least one of analyze and process a particular node;

systematically walking through nodes within the ~~tree~~ graph;

generating an encounter event and handling the encounter event with a bound node observer when one of the node patterns matches an attribute of an encountered node;

deactivating the ~~tree~~ graph observer associated with the bound node observer if the bound node observer determines that a set of sub-nodes of the encountered node should be pruned; and

bypassing the walking of the set of sub-nodes without deleting the set of sub-nodes if all of the plurality of ~~tree~~ graph observers have been deactivated.

13. (Currently Amended) The method of claim 12, comprising the further step of generating a completed event for each node observer that received an encountered event and that did not cause the ~~tree~~ graph observer to become deactivated.

14. (Currently Amended) The method of claim 12, comprising the further step of reactivating the ~~tree~~ graph observer associated with the bound node observer after the set of sub-nodes of the encountered node have been walked.

15. (Currently Amended) The method of claim 12, comprising the further step of reactivating the ~~tree~~ graph observer associated with the bound node observer after set of sub-nodes of the encountered node have been bypassed.

16. (Currently Amended) The method of claim 12, comprising the further step of walking the sub-nodes if at least one ~~tree~~ graph observer is active.

17. (Currently Amended) A program product stored on a recordable medium, which when executed, analyzes a ~~tree~~ directed non-cyclical graph of hierarchical data, the program product comprising:

program code configured to bind a plurality of tree observers that look for matching node patterns to a ~~tree~~ graph, wherein each ~~tree~~ graph observer is further bound to a set of node patterns that identify distinguishing node attributes and a set of node observers that at least one of analyze and process a particular node;

program code configured to provide ~~tree~~ graph walking logic for systematically walking through nodes within the ~~tree~~ graph;

program code configured to provide a first pruning system that can be instructed by a node observer bound with an associated ~~tree~~ graph observer to deactivate the associated ~~tree~~ graph observer until a set of sub-nodes for an encountered node has been walked; and

program code configured to provide a second pruning system that can instruct the ~~tree~~ graph walking logic not to walk the set of sub-nodes for the encountered node without deleting the set of sub-nodes.

18. (Currently Amended) The program product claim 17, wherein the second pruning system will cause the set of sub-nodes not to be walked only if all of the plurality of ~~tree~~ graph observers have been deactivated.

19. (Original) The program product claim 17, further comprising program code configured to provide a pattern testing system for determining if the encountered node matches one of the node patterns.

20. (Original) The program product claim 17, further comprising program code configured to provide an event manager for generating an encountered event when one of the node observers is bound to a matching node pattern.